Package ‘PoisNonNor’

March 10, 2018

Type Package

Title Simultaneous Generation of Count and Continuous Data

Version 1.5

Date 2018-03-10

Author Hakan Demirtas, Yaru Shi, Rawan Allozi

Maintainer Rawan Allozi <ralloz2@uic.edu>

Description Generation of count (assuming Poisson distribution) and continuous data (using Fleishman polynomials) simultaneously.

License GPL-2 | GPL-3

Depends graphics, grDevices, stats, utils, BB, corpcor, MASS, Matrix

NeedsCompilation no

Repository CRAN

Date/Publication 2018-03-10 21:19:23 UTC

R topics documented:

PoisNonNor-package ........................................ 2
bounds.corr.GSC.NN .................................... 3
bounds.corr.GSC.NNP ................................... 4
bounds.corr.GSC.PP .................................... 5
fleishman.roots .......................................... 6
intercor.all ........................................... 7
intercor.NN ............................................... 8
intercor.NNP ......................................... 9
intercor.PP ............................................ 10
Param.fleishman ......................................... 11
RNG_P_NN ............................................. 12
Validate.correlation .................................. 13

Index 15
PoisNonNor-package

Simultaneous generation of count and continuous data with Poisson and continuous marginals

Description

A package for simulating multivariate data with count and continuous variables with a pre-specified correlation matrix and marginal distribution. The count variables are assumed to have Poisson distribution, and continuous variables can take any shape that is allowed by the Fleishman polynomials. This mixed data generation scheme is a combination of the normal to anything principle for the count part, and multivariate continuous data generation mechanism via the Fleishman polynomials.

Details

Package: PoisNonNor
Type: Package
Version: 1.5
Date: 2018-03-10
License: GPL-2 | GPL-3

This package consists of eleven functions. The functions bounds.corr.GSC.NN, bounds.corr.GSC.NNP, and bounds.corr.GSC.PP return the lower and upper bounds of the pairwise correlation of continuous-continuous, continuous-count, and count-count pairs, respectively. The function Validate.correlation validates the specified quantities to avoid obvious correlation matrix specification errors in regarding to the correlation matrix. The functions intercor.NN, intercor.NNP, and intercor.PP give the intermediate normal correlation matrix for continuous-continuous, continuous-count, and count-count combinations, respectively. The function intercor.all returns the final intermediate correlation matrix by combining the three parts of correlation together. The function Param.fleishman calculates the Fleishman coefficient. The engine function RNG_P_NN generates mixed data in accordance with the specified marginal and correlation matrix.

n1, n2, and n=n1+n2 stand for the number of count, continuous, and the total number of the variables, respectively. By design, the first n1 variables are count, and the last n2 variables are continuous in the generated data matrix.

Author(s)

Hakan Demirtas, Yaru Shi, Rawan Allozi
Maintainer: Rawan Allozi <ralloz2@uic.edu>

References

bounds.corr.GSC.NN


**bounds.corr.GSC.NN**

*Computes the approximate lower and upper bounds of the correlation matrix entries for the continuous pairs*

**Description**

This function calculates the approximate lower and upper bounds for all continuous pairs by the method in Demirtas and Hedeker (2011).

**Usage**

```r
bounds.corr.GSC.NN(pmat)
```

**Arguments**

- **pmat**: a n2x4 matrix where each row includes the four coefficients (a,b,c,d) of the Fleishman’s system.

**Details**

The approximate correlation bounds are computed via the 'Generate, Sort and Correlate' (GSC) technique, proposed by Demirtas and Hedeker (2011).

**Value**

Returns a list with two components

- **min**: lower correlation bound matrix
- **max**: upper correlation bound matrix

**References**


**See Also**

bounds.corr.GSC.NNP, bounds.corr.GSC.PP
Examples

```r
## Not run:
pmat = matrix(c(
  0.1148643, 1.0899150, -0.1148643, -0.0356926,
  -0.0488138, 0.9203374, 0.0488138, 0.0251256,
  -0.2107427, 1.0398224, 0.2107427, -0.0293247), nrow=3, byrow=TRUE)

bounds.corr.GSC.NNP (pmat)

## End(Not run)
```

---

`bounds.corr.GSC.NNP`  *Computes the approximate lower and upper bounds of the correlation matrix entries for the continuous-count pairs*

Description

This function calculates the approximate lower and upper bounds for all continuous-count pairs by the method in Demirtas and Hedeker (2011).

Usage

`bounds.corr.GSC.NNP(lamvec, pmat)`

Arguments

- `lamvec`  a vector of lambda values of length `n1`.
- `pmat`  a `n2x4` matrix where each row includes the four coefficients (a,b,c,d) of the Fleishman’s system.

Details

The approximate correlation bounds are computed via the ‘Generate, Sort and Correlate’ (GSC) technique, proposed by Demirtas and Hedeker (2011).

Value

Returns a list with two components, both are matrices of size `n1x2` where `n1` and `n2` are the number of count and continuous variables, respectively.

- `min`  lower correlation bound matrix
- `max`  upper correlation bound matrix

References

See Also

bounds.corr.GSC.NN, bounds.corr.GSC.PP

Examples

```r
## Not run:
pmat = matrix(c(
  0.1148643, 1.0899150, -0.1148643, -0.0356926,
  -0.0488138, 0.9203374, 0.0488138, 0.0251256,
  -0.2107427, 1.0398224, 0.2107427, -0.0293247), nrow=3, byrow=TRUE)

lamvec = c(0.5, 0.7, 0.9)

bounds.corr.GSC.NN(pmat, lamvec)

## End(Not run)
```

```

bounds.corr.GSC.PP

Computes the approximate lower and upper bounds of the correlation matrix entries for the count pairs

Description

This function calculates the approximate lower and upper bounds for all count pairs by the method in Demirtas and Hedeker (2011).

Usage

```r
bounds.corr.GSC.PP(lamvec)
```

Arguments

```r
lamvec a vector of lambda values of length n1.
```

Details

The approximate correlation bounds are computed via the 'Generate, Sort and Correlate' (GSC) technique, proposed by Demirtas and Hedeker (2011).

Value

Returns a list with two components, both are matrices of size n1 x n1.

```r
min lower correlation bound matrix
max upper correlation bound matrix
```
fleishman.roots

References


See Also

bounds.corr.GSC.NNP, bounds.corr.GSC.PP

Examples

## Not run:
lamvec = c(0.5, 0.7, 0.9)

bounds.corr.GSC.PP(lamvec)

## End(Not run)

---

fleishman.roots An auxiliary function that is called by Param.fleishman function

Description

This function sets up formulae that are needed at the subsequent stages.

Usage

fleishman.roots(p, r)

Arguments

p a vector of length three that contains the Fleishman coefficients.

r a vector of length two that contains skewness and kurtosis values.

References


See Also

Param.fleishman
**intercor.all**

*Computes the intermediate correlation matrix*

**Description**

This function computes the intermediate correlation matrix of the multivariate normal distribution that provides a basis for subsequent transformations.

**Usage**

`intercor.all(cmat, pmat, lamvec)`

**Arguments**

- `cmat` a \((n1+n2)\times(n1+n2)\) matrix of specified correlations.
- `pmat` a \(n2\times4\) matrix where each row includes the four coefficients \((a,b,c,d)\) of the Fleishman’s system.
- `lamvec` a vector of lambda values of length \(n1\).

**Details**

This function assembles all three submatrices that are pertinent to all continuous-continuous, count-count, and count-continuous pairs.

**Value**

Returns an intermediate matrix of size \((n1+n2)\times(n1+n2)\).

**See Also**

`intercor.NN`, `intercor.NNP`, `intercor.PP`

**Examples**

```r
## Not run:
pmat = matrix(c(
  0.1148643, 1.0899150, -0.1148643, -0.0356926,
  -0.0488138, 0.9283374, 0.0488138, 0.0251256,
  -0.2107427, 1.0398224, 0.2107427, -0.0293247), nrow=3, byrow=TRUE)

lamvec = c(0.5, 0.7, 0.9)

cmat = matrix(c(
  1.000, 0.352, 0.265, 0.342, 0.090, 0.141,
  0.352, 1.000, 0.121, 0.297, -0.022, 0.177,
  0.265, 0.121, 1.000, 0.294, -0.044, 0.129,
  0.342, 0.297, 0.294, 1.000, 0.100, 0.354,
  0.090, -0.022, -0.044, 0.100, 1.000, 0.386,
  0.141, 0.177, 0.129, 0.354, 0.386, 1.000),
```
intercor.NN Computes the subset of the intermediate correlation matrix that is pertinent to the continuous pairs

Description

This function computes the submatrix of the intermediate correlation matrix of the multivariate normal distribution. It is relevant to the continuous part of the data.

Usage

intercor.NN(pmat, cmat)

Arguments

pmat a n2x4 matrix where each row includes the four coefficients (a,b,c,d) of the Fleishman’s system.

cmat a n2xn2 matrix of specified correlations for the continuous part.

Details

Fleishman polynomials is a method to generate real-life non-normal distributions of variables by using their first four moments. It is based on the polynomial transformation, \( Y = a + bZ + cZ^2 + dZ^3 \), where \( Z \) follows a standard normal distribution and \( Y \) is standardized (zero mean and unit variance).

Normal-Normal correlation for a given continuous pair can be calculated by solving the following equation.

\[
\begin{align*}
    r_{y_1y_2} &= r_{z_1z_2}(b_1b_2 + 3b_1d_2 + 3d_1b_2 + 9d_1d_2) + r_{z_1z_2}^2(2c_1c_2) + r_{z_1z_2}^3(6d_1d_2)
\end{align*}
\]

Value

Returns an intermediate matrix of size n2xn2

References

Examples

```r
## Not run:
pmat = matrix(c(0.1148643, 1.0899150, -0.1148643, -0.0356926,
                -0.0488138, 0.9203374, 0.0488138, 0.0251256,
                -0.2107427, 1.0398224, 0.2107427, -0.0293247), nrow=3, byrow=TRUE)
cmat = matrix(c(1.000, 0.100, 0.354,
                0.100, 1.000, 0.386,
                0.354, 0.386, 1.000), nrow=3, byrow=TRUE)

intercor.NNP(pmat, cmat)

## End(Not run)
```

---

`intercor.NNP`  
*Computes the subset of the intermediate correlation matrix that is pertinent to the count-continuous pairs*

Description

This function computes the submatrix of the intermediate correlation matrix of the multivariate normal distribution. It is relevant to the count-continuous part of the data.

Usage

```r
intercor.NNP(lamvec, cmat, pmat)
```

Arguments

- `lamvec`: a vector of lambda values of length `n1`.
- `mat`: a `(n1+n2)x(n1+n2)` matrix of specified correlations.
- `pmat`: a `n2x4` matrix where each row includes the four coefficients (a,b,c,d) of the Fleishman’s system.

Details

Calculations are done by combining the methods described in Demirtas, Hedeker and Mermelstein (2012) and Amatya and Demirtas (2017).

Value

Returns an intermediate correlation matrix of size `n1 x n2`
References


Examples

```r
## Not run:
pmat = matrix(c(
  0.1148643, 1.0899150, -0.1148643, -0.0356926,
  -0.0488138, 0.9203374, 0.0488138, 0.0251256,
  -0.2107427, 1.0398224, 0.2107427, -0.0293247), nrow=3, byrow=TRUE)

lamvec = c(0.5, 0.7, 0.9)

cmat = matrix(c(
  0.342, 0.090, 0.141,
  0.297, -0.022, 0.177,
  0.294, -0.044, 0.129), nrow=3, byrow=TRUE)

intercor.NN(lamvec, cmat, pmat)

## End(Not run)
```

```r
intercor.PP  Computes the subset of the intermediate correlation matrix that is pertinent to the count pairs
```

Description

This function computes the submatrix of the intermediate correlation matrix of the multivariate normal distribution. It is relevant to the count part of the data.

Usage

```r
intercor.PP(lamvec, cmat)
```

Arguments

- `lamvec`: a vector of lambda values of length n1.
- `cmat`: a n1xn1 matrix of specified correlations.

Details

Calculations are done by combining the methods described in Yahav and Shumeli (2012) and Amatya and Demirtas (2017).
**Param.fleishman**

*Calculates the Fleishman coefficients*

**Description**

This function calculates the four coefficients in the Fleishman system given skewness and kurtosis values.

**Usage**

```r
Param.fleishman(rmat)
```

**Arguments**

- `rmat` a n2x2 matrix that includes skewness and kurtosis values for each continuous variable, where the first and second columns represent skewness and kurtosis, respectively.

**Value**

Returns a matrix of size n2x4 where rows and columns represent variables and coefficients, respectively.

**References**


**Examples**

```r
## Not run:
lamvec = c(0.5, 0.7, 0.9)
cmat = matrix(c(
  1.000, 0.352, 0.265,
  0.352, 1.000, 0.121,
  0.265, 0.121, 1.000), nrow=3, byrow=TRUE)
tercor.PP(lamvec, cmat)
## End(Not run)
```
RNG_P_NN

**References**


**Examples**

```r
## Not run:
rmatrix = matrix(c(-0.5486, -0.2103, 0.3386, 0.9035, 1.0283, 0.9272), byrow=TRUE, ncol=2)
Param.fleishman(rmatrix)

## End(Not run)
```

---

**RNG_P_NN**

*Simultaneously generates count and continuous data*

**Description**

This function simulates count and continuous data, where the count part is assumed to follow a multivariate Poisson distribution and the continuous part can take any shape allowed by the Fleishman polynomials. A correlation matrix and marginal features (rate parameter for the count variables, and skewness and kurtosis parameters for the continuous variables must be supplied by users).

**Usage**

```r
RNG_P_NN(lamvec, cmat, rmat, norow, meanvec, variancevec)
```

**Arguments**

- `lamvec`: a vector of lambda values of length n1
- `cmat`: specified correlation matrix
- `rmat`: a n2x2 matrix that includes skewness and kurtosis values for each continuous variable
- `norow`: number of rows in the multivariate mixed data
- `meanvec`: mean vector for continuous variables of length n2
- `variancevec`: variance vector for continuous variables of length n2

**Value**

Returns a data matrix of size norowx(n1+n2). By design, the first n1 variables are count, and the last n2 variables are continuous.
Validate.correlation

References


Examples

```r
## Not run:
lamvec = c(0.5, 0.7, 0.9)

cmat = matrix(c(  
1.000, 0.352, 0.265, 0.342, 0.090, 0.141,  
0.352, 1.000, 0.121, 0.297, -0.022, 0.177,  
0.265, 0.121, 1.000, 0.294, -0.044, 0.129,  
0.342, 0.297, 0.294, 1.000, 0.100, 0.354,  
0.090, -0.022, -0.044, 0.100, 1.000, 0.386,  
0.141, 0.177, 0.129, 0.354, 0.386, 1.000), nrow=6, byrow=TRUE)

rmat = matrix(c(-0.5486, -0.2103, 0.3386, 0.9035, 1.0283, 0.9272), byrow=TRUE, ncol=2)

norow=10e+5

mean.vec=c(1, 0.5, 100)

variance.vec=c(1, 0.02777778, 1000)

P_NN_data = RNG_P_NN(lamvec, cmat, rmat, norow, mean.vec, variance.vec)

## End(Not run)
```

Validate.correlation  Checks the validity of the specified correlation matrix

Description
The function checks the validity of pairwise correlations. Additionally, it checks positive definiteness, symmetry and correctness of the dimensions.
Validate.correlation

Usage

Validate.correlation(cmat, pmat = NULL, lamvec = NULL)

Arguments

- **cmat**: a nxn matrix of specified correlations for the n-variate distribution.
- **pmat**: a n2x4 matrix where each row includes the four coefficients (a,b,c,d) of the Fleishman’s system.
- **lamvec**: a vector of lambda values of length n1.

Details

In addition to being positive definite and symmetric, the values of pairwise correlations in the target correlation matrix must also fall within the limits imposed by the marginal distributions in the system. The function ensures that the supplied correlation matrix is valid for simulation. If a violation occurs, an error message is displayed that identifies the violation. The function returns a logical value TRUE when no such violation occurs.

See Also

- bounds.corr.GSC.PP
- bounds.corr.GSC.NN
- bounds.corr.GSC.NNP

Examples

```r
## Not run:
pmat = matrix(c(
  0.1148643, 1.0899150, -0.1148643, -0.0356926,
  -0.0488138, 0.9203374, 0.0488138, 0.0251256,
  -0.2107427, 1.0398224, 0.2107427, -0.0293247), nrow=3, byrow=TRUE)
lamvec = c(0.5,0.7,0.9)

mat = matrix(c(
  1.000, 0.352, 0.265, 0.342, 0.090, 0.141,
  0.352, 1.000, 0.121, 0.297, -0.022, 0.177,
  0.265, 0.121, 1.000, 0.294, -0.044, 0.129,
  0.342, 0.297, 0.294, 1.000, 0.100, 0.354,
  0.090, -0.022, -0.044, 0.100, 1.000, 0.386,
  0.141, 0.177, 0.129, 0.354, 0.386, 1.000), nrow=6, byrow=TRUE)

Validate.correlation (mat,pmat,lamvec)

## End(Not run)
```
Index

bounds.corr.GSC.NN, 2, 3, 5, 14
bounds.corr.GSC.NNP, 2, 3, 4, 6, 14
bounds.corr.GSC.PP, 2, 3, 5, 5, 6, 14

fleishman.roots, 6

intercor.all, 2, 7
intercor.all (intercor.all), 7
intercor.NN, 2, 7, 8
intercor.NNP, 2, 7, 9
intercor.PP, 2, 7, 10

Param.fleishman, 2, 6, 11
PoisNonNor (PoisNonNor-package), 2
PoisNonNor-package, 2

RNG_P_NN, 2, 12

Validate.correlation, 2, 13